Dart 3

1. Hello World

void main() {

print('Hello, World!');

}

Her uygulama main fonksiyonu ile başlar. Bu yüzden bir main fonksiyonu oluşturarak başlıyoruz. Void ise bu fonksiyonun bir geri dönüşü olmadığını belirtir.

Süslü parantezler içerisine yazılan kodlar main fonksiyonuna dahildir.

Main fonksiyonu çalıştırıldığında yani program başlatıldığında sadece süslü parantez içerisinde olan kısım çalışır.

Konsolda yazı yazdırmak için print fonksiyonu kullanılır. Print fonksiyonu içerisinde yazılan ifade ile program çalıştırılınca içerisindeki yazı konsolda görünür.

Kod satırı yazıldıktan sonra ; ile satırın bitirildiği belirtilmelidir.

Bu sayede ilk programımızı çalıştırarak konsoldaki ilk değerimizi görmüş olduk.

1. var

Var ile değişken tanımlanır. Var ile bir tür belirtmeye gerek kalmaz. Önce var anahtarı yazılır daha sonra değişken ismi tanımlanır. Eşittir ibaresinden sonra da değişkenin değeri belirtilir. Bu değer sayı olabilir. Yazı olabilir. Ondalıklı yazı olabilir. İstenilen her şey olabilir.

Var name = ‘Hello, World!’

void main() {

print(name);

}

Artık değişken üzerinden direkt olarak değerimizi çağırabiliriz.

Şimdi değişkenlere neden ihtiyacımız var düşünmeye çalışalım. Örneğin alt alta 4 tane hello world yazmak isteyelim. Bu şekilde yazabiliriz ancak olur da hepsinin sonuna ünlem işareti yerine ? işareti eklememiz gereksin. Bu durumda her birini sırayla değiştirmek gerekir. Örneğin

void main() {

print('Hello, World!');

print('Hello, World!');

print('Hello, World!');

print('Hello, World!');

}

Ancak bunun yerine değişkene değeri atayarak daha kolay bir şekilde değiştirebiliriz.

Örneğin

Var hello = ‘Hello, World!’

Seklinde bir değişken tanımlayalım. Artık Hello, World yazan her yerde hello değişkenini direkt olarak kullanabiliriz.

void main() {

Var hello = ‘Hello, World!’

print(hello);

print(hello);

print(hello);

print(hello);

}

Bu sayede olur da yeni bir değişiklik yapmak istersek direkt olarak hello değişkeni üzerinde değişiklik yaparak hepsinin değişmesini sağlayabiliriz.

Bir de yazı yerine sayı ile deneyelim. Örneğin kullanıcının yaşını 5 kez yazdıralım ve sonra yaşın değişmesi gereksin. Bu durumda da aynı yolu izleyebiliriz.

void main() {

print(20);

print(20);

print(20);

print(20);

}

Bu şekilde kullanmak yerine değişkene atayalım ve tekrar kullanalim.

void main() {

Var yas = 20;

print(yas);

print(yas);

print(yas);

print(yas);

}

Bu sayede yaş değerini değiştirmek istediğimizde direkt olarak değiştirebilmiş olduk. Değişkenlerin neden kullanılması gerektiğine dair temel özelliği öğrendiğimize göre bir sonraki derste de değişkenlerin türlerinin neler olduğunu öğrenelim.

Klasör mimarisi

Bin: Dart projesinin çalıştırılabilir dosyalarını içeren klasördür. Bu klasör genellikle projeye doğrudan ilişkili olan ana uygulama dosyasını içerir.

Lib: Dart projesinin kütüphane kodlarının içeren klasördür. Burada projenin geri kalanında kullanılan sınıflar, fonksiyonlar ve diğer yardımcı kodlar bulunur.

.dart\_tool: Dart projeleri için otomatik oluşturulan bir klasördür. Bu klasör dart araçları tarafından kullanılan araçlar, geçici dosyalar ve diğer projeye özgü bilgiler gibi çeşitli dosyalar içerir. Bu klasörün doğrudan değiştirilmesi veya silinmesi gerekmez.

Test: Yazılım geliştirme sürecinde testlerin yerleştirildiği bir klasördür. Bu klasörde projenin birim testlerini, entegrasyon testlerini veya kabul testlerini içeren test kodlarının barındırır. Test klasörü projenin kaynak kodundan ayrı tutulur ve genellikle proje yapısal olarak ayrı bir bölüm olarak yönetilir.

.gitignore: Git deposunda yer alan dosyaların takip edilmemesi gerektiğini söyler.

Analysis\_options.yaml: Dart dilinde statik analiz ayarlarının yapılandırıldığı bir dosyadır. Bu dosya dart kodunun analizini yapılandırmak ve analiz araçlarının nasıl çalışacağını belirlemek için kullanılır. Bu sayede kod kalitesini artırmak, hataları tespit etmek, okunabilirliği ve sürdürülebiliriği iyileştirmek için kullanılır.

Changelog.md: yazılım geliştirme sürecinde yapılan değişiklikleri ve sürüm notlarını kaydetmek için kullanılan bir markdown dosyasıdır. Bu dosya ile projenin farklı sürümleri arasındaki değişiklikleri takip etmek, kullanıcılara yeni özellikleri, düzeltmeleri ve iyileştirmeleri sunmak amacıyla oluşturulur.

pubspec.lock: dart dilinceki paket bağımlılıklarını ve kullanılan sürümlerinin tam bir kaydının tutan bir dosyadır.

pubspec.yaml: dart dilinceki bir proje için bağımlılıkları versiyonları, isimleri ve diğer proje ayarlarının tanımlayan bir konfigürsayon dosyasıdır. Paket yönetimi, kütüphane ve modülleri eklemek için kullanılır.

readme.md: Projenin açıklamasını, kullanım talimatini, gereksinimleri ve diğer önemli bilgileri içerir.